图文 107 我们系统中使用的Producer是如何创建出来的?

详情 评论

我们系统中使用的Producer是如何创建出来的?

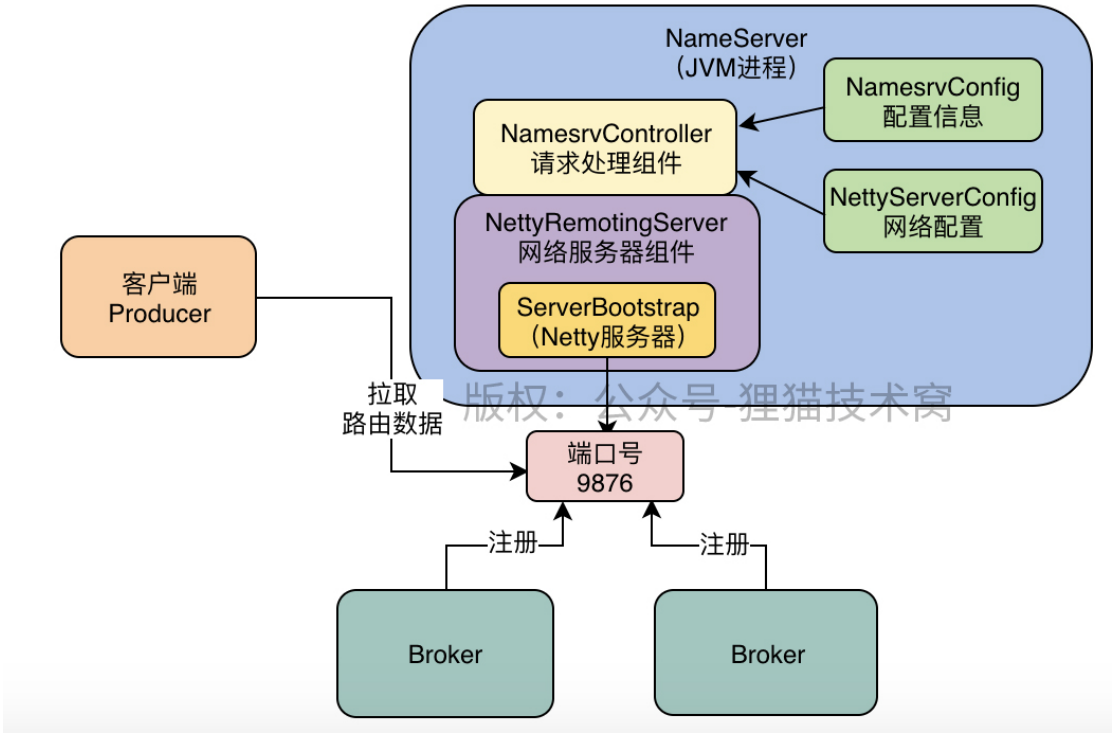
继《从零开始带你成为JVM实战高手》后，阿里资深技术专家携新作再度出山，重磅推荐：

（点击下方蓝字试听）

《从零开始带你成为MySQL实战优化高手》

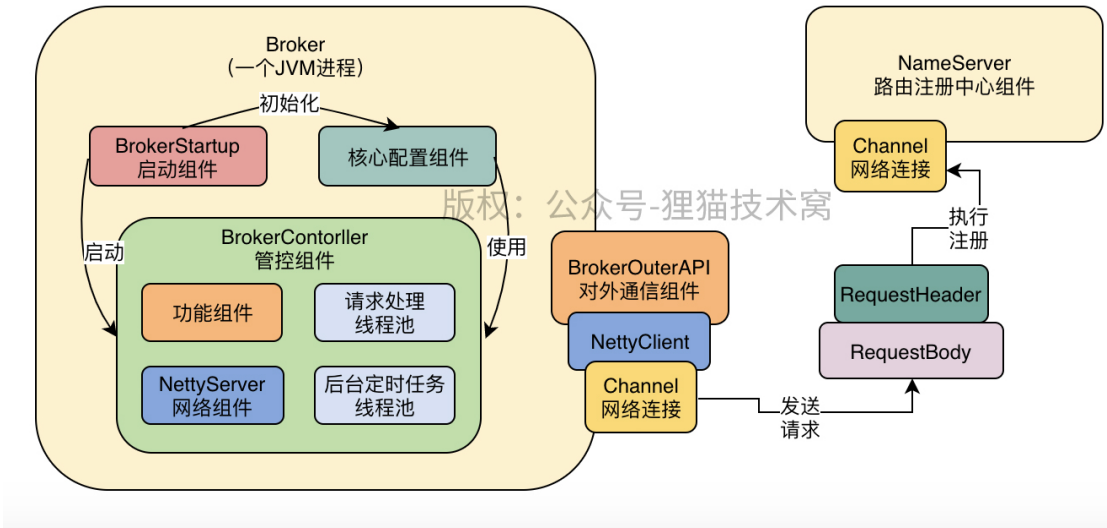
在之前的学习中，我们已经讲完了NameServer启动流程的相关源码，同时也分析出了NameServer启动之后的核心架构，如下图所示

大家可以看一下回顾一下，一定要记得，牢牢地抓住他里面的一些核心组件。



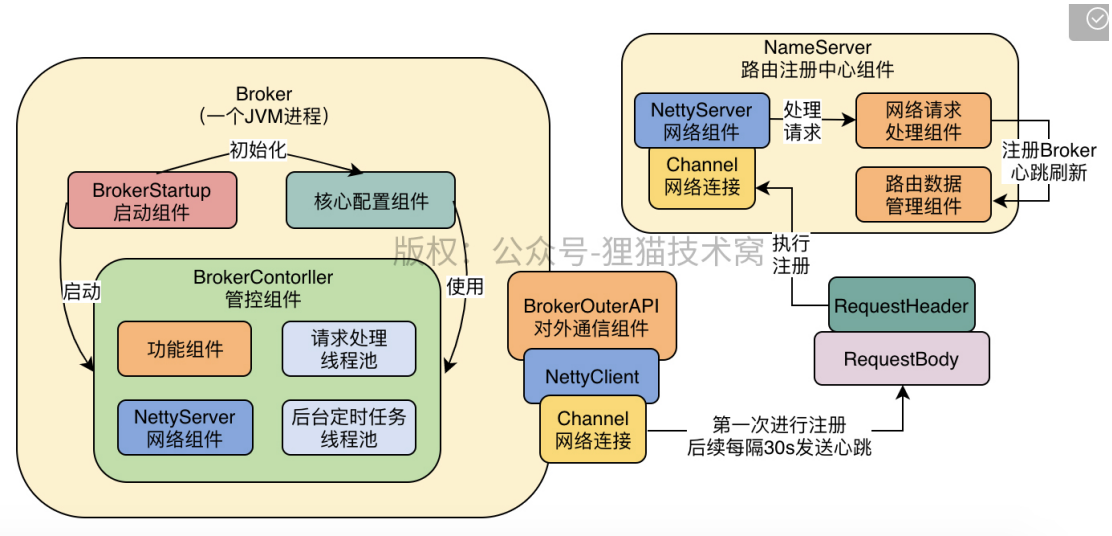
简单来说，NameServer启动之后，就会有一个核心的NamesrvController组件，他就是用于控制NameServer的所有行为的，包括内部启动一个Netty服务器去监听一个9876端口号，然后接收处理Broker和客户端发送过来的请求。

接着我们还学习了Broker启动过程的相关源码，也分析出了Broker启动之后的核心架构，我们如下图所示。



简单来说，Broker启动之后，最核心的就是有一个BrokerController组件管控Broker的整体行为，包括初始化自己的Netty服务器用于接收客户端的网络请求，包括启动处理请求的线程池、执行定时任务的线程池，初始化核心功能组件，同时还会启动之后发送注册请求到NameServer去注册自己。

同时我们之前还讲解完了Broker启动之后进行注册以及定时发送注册请求作为心跳的机制，以及NameServer有一个后台进程定时检查每个Broker的最近一次心跳时间，如果长时间没心跳就认为Broker已经故障，我们看下图。



其实大家要知道一点，在讲完这些内容过后，你可以认为在我们的RocketMQ集群里，已经启动好了NameServer，而且还启动了一批Broker，同时Broker都已经把自己注册到NameServer里去了，NameServer也会去检查这批Broker是否存活。

其实此时我们不需要去关注NameServer和Broker干了别的什么事情，这个时候我们只要知道已经有了一个可用的RocketMQ集群就可以了，然后此时我们是不是就可以让自己开发好的系统去发送消息到MQ里去了？

没错，所以此时我们就需要引入一个Producer组件了，实际上，大家要知道，我们开发好的系统，最终都是要构建一个Producer组件，然后通过Producer去发送消息到MQ的Broker上去的

所以今天开始我们就来讲一下Producer这个组件的底层原理，当然先是得从Producer的构造开始了

既然要说Producer的构造，那肯定是要先回顾一下Producer是如何构造出来的，其实我们可以回顾一下下面的这块使用Producer发送消息到MQ的代码，就能清晰的看到Producer是如何构造出来的。

```
DefaultMQProducer producer = new DefaultMQProducer("order_producer_group");

producer.setNamesrvAddr("localhost:9876");

producer.start();
```

大家可以看到，其实构造Producer很简单，就是创建一个DefaultMQProducer对象实例，在其中传入你所属的Producer分组，然后设置一下NameServer的地址，最后调用他的start()方法，启动这个Producer就可以了。

其实创建DefaultMQProducer对象实例是一个非常简单的过程，无非就是创建这么一个对象出来，然后保存一下他的Producer分组。设置NameServer地址也是一个很简单的过程，无非就是保存一下NameServer地址罢了。

其实最核心的还是调用了这个DefaultMQProducer的start()方法去启动了这个消息生产组件，那么这个start()都干了什么呢？

这个我们下周继续讲解，今天就作为一个承上启下的过程，大家知道我们目前对RocketMQ底层原理剖析到了哪个阶段，接下去要看哪个阶段就可以了。

End